CS-308-2014 Final Report

Smart Home

Gladiators

TH-07

Rahul Gupta, 100050086
Prakhar Jain, 100050024
Utkarsh Diwaker, 100050087
Saket Tiwari, 100050075

Table of Contents

1. Introduction

The system we developed has come to be called as Smart Home collectively, including the hardware and software components. The motivation for developing Smart Home was to provide a central control dock for all home switches and appliances. A dock like this would only have utility if it could be accessed from anywhere, be it a remote location or locally inside the house.

The Smart Home system allows access to switches through a web page which can be accessed from anywhere with an internet connection. A user can view the state of a switch in real time as it changes. In addition to that accessing sensor values with a delay of two seconds is also available on this system. Control of switches depending on the sensor values has also been enabled. This kind of a setup can be useful in all environments requiring remote and intelligent control of electrical switches, which can range from homes to offices and can even be applied to a greenhouse.

In this report first we give an overall description of the Smart Home. Next we go into specific details of hardware, software, user interface, communication interface and testing strategies in section 3. We have explained the intricacies of each of these topics in the corresponding sections.

2. Problem Statement

We intended to come up with a system which would allow us to control the switches of our home remotely via a web interface. The system should also be smart enough to automatically turn on/off the switches depending upon the sensors in the room.

We believe that this is a real life problem that have moved to using smaller portable devices connected to the internet.The product makes the users life easier by not requiring him/her to be physically present at a place to switch on/off the electrical appliances and fixtures. This product brings switches to the user's hand.

With the advent of mobile technology people the tech savvy users to remotely control their switches. The functioning and efficacy of the product depends on the internet connectivity.

3. Requirements

3.1 Functional Requirements

The functional requirements of our project are as follows -

1. Switches on/off in place
2. Remote control of switches via web
3. LED lights for in place switches
4. Interactive user interface
5. Any changes made to the switch in place should be reflected on the web in real time.

3.2 Hardware Requirements

1. **Raspberry Pi -** A centralised server for a home/greenhouse which communicates with the arduino boards through XBee.
2. **Arduino Boards -** We use Arduino Duo as hardware interfaces that control the switches. These are connected to the Raspberry Pi over wifi via XBee chips.
3. **XBee -** XBees are used for communications between Raspberry Pi and Arduino boards.
4. **Switchboard -** Switches are attached on the switchboard to manually turn on/off LEDs. These switchboards are connected to their respective Arduino.

3.3 Software Requirements

1. **Arduino IDE -** This software is required to program the Arduino Boards according to our requirement.
2. **NodeJS Web Server -** We are using the NodeJS Web server to create persistent web sockets between the web browser and the Raspberry Pi. This helps us to have real time communication between them.
3. **MySQL -** MySQL is used as a database management system for storing the current status of switches in the database. It is used by NodeJS web server.

4. System Design

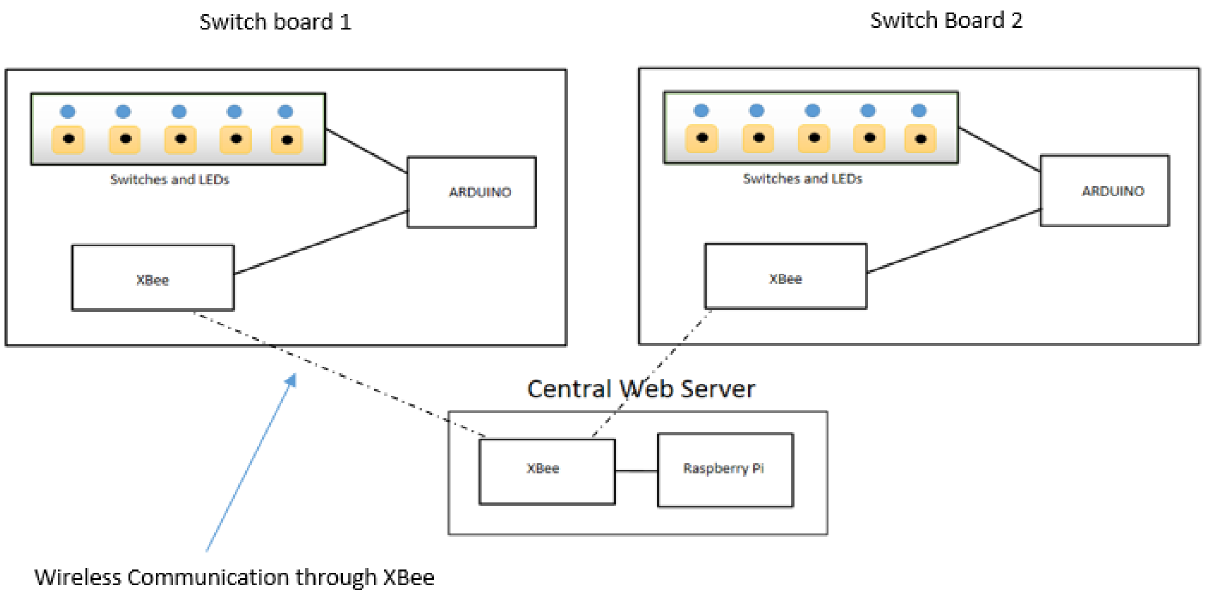


Figure 1: System Design

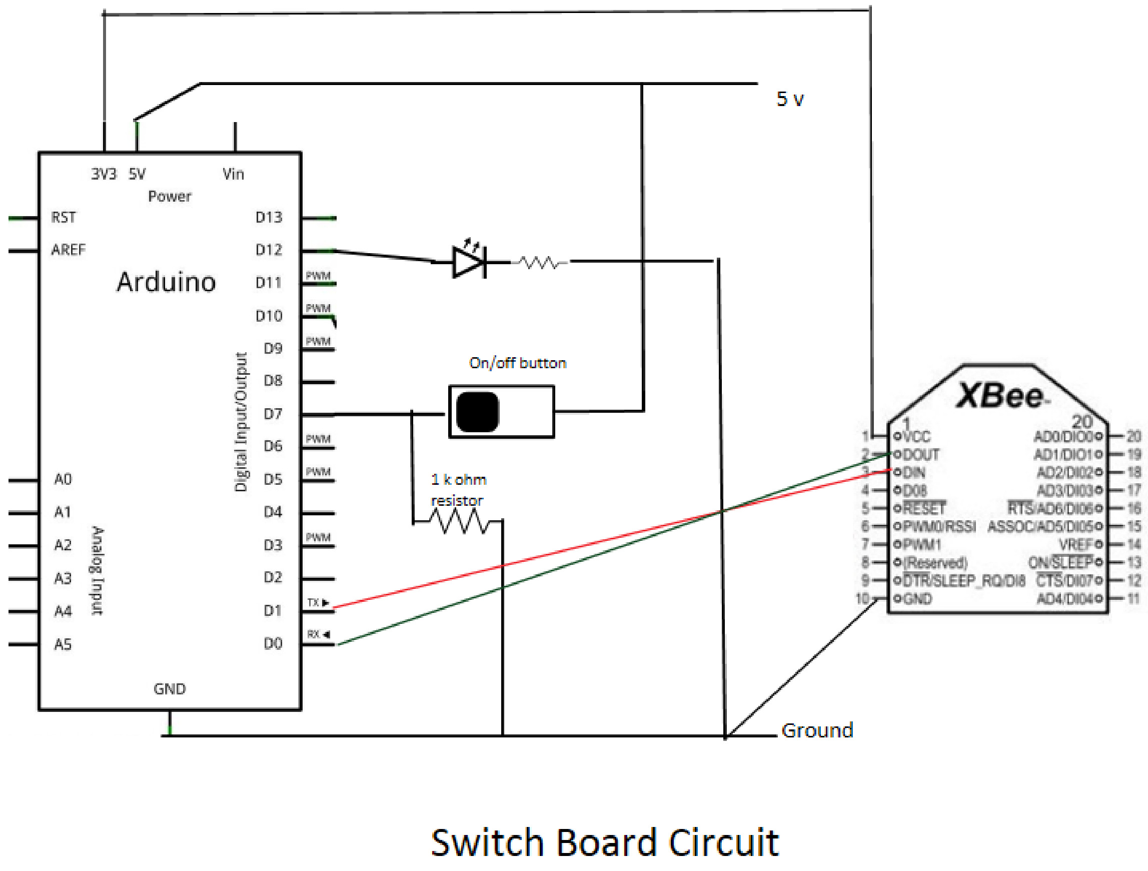
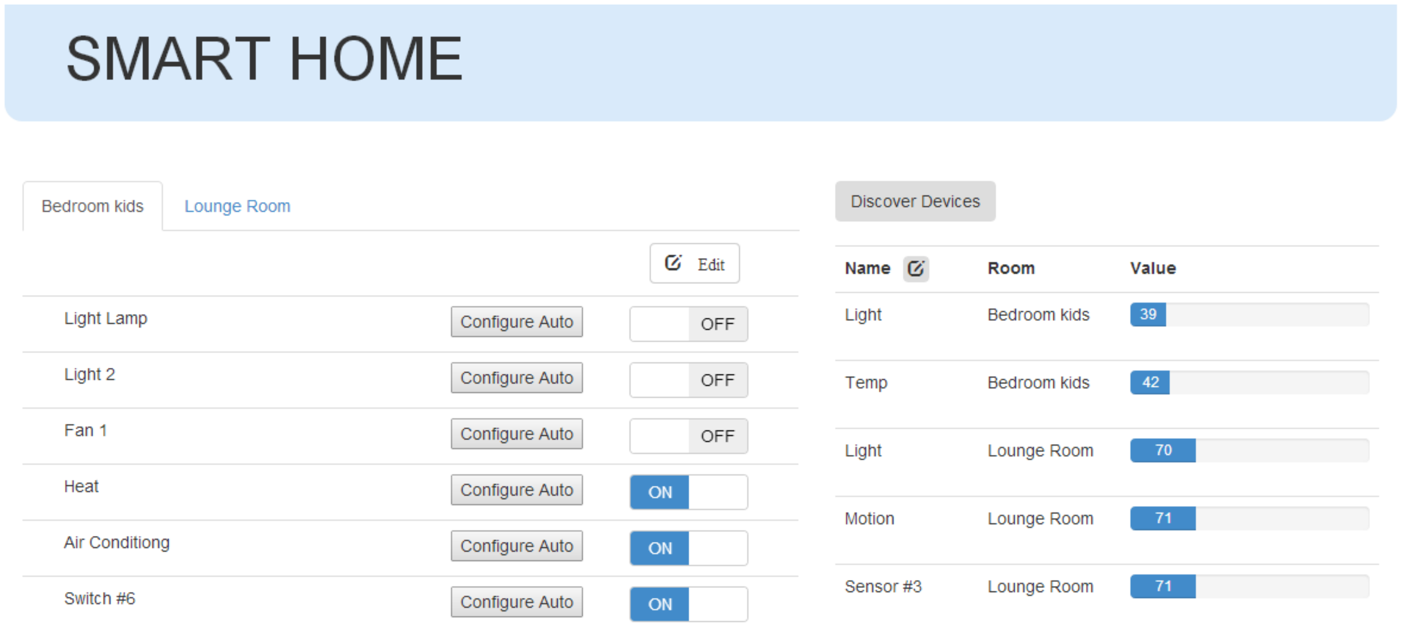


Figure 2: Switch-board Circuit

1. There will be a single Raspberry Pi hosting a central web server in the entire home. This is shown in Figure 1.
2. Each switch board in the home comprises of an Arduino board to control the switches, a XBee connected to the Arduino which is used to communicate with the Raspberry Pi. A detailed switch-board circuit is shown in Figure 2.
3. Different type of sensors attached to the Arduino are used to get the sensor values which are then sent to the central web server (through XBee). The central web server analyzes these values and turns the switches on/off accordingly.
4. A Web Interface, which allows the user to remotely switch on/off the appliances (LEDs in our case). It also displays the current state (on/off) of each switch and sensor values of each sensor in the home. It also provides an option to control the switches based on sensor values. This is shown in Figure 3. So, In Figure 3, “Bedroom Kids” is one switch-board and “Lounge Room” is another switch-board.


SMART HOME

Bedroom kids | Lounge Room

Edit

Light Lamp	Configure Auto	OFF
Light 2	Configure Auto	OFF
Fan 1	Configure Auto	OFF
Heat	Configure Auto	ON
Air Conditiong	Configure Auto	ON
Switch #6	Configure Auto	ON

Discover Devices

Name	Room	Value
Light	Bedroom kids	39
Temp	Bedroom kids	42
Light	Lounge Room	70
Motion	Lounge Room	71
Sensor #3	Lounge Room	71

Figure 3: Web Interface

5. Working of the System and Test results

5.1 Communication between Arduino and Raspberry Pi

Messages are passed from Arduino to Raspberry Pi server or vice-versa via XBee. We have made a custom protocol for supporting different types of messages. The message is of the form:

255 <switch_board_id> <Message_Type> <Field_A> <Field_B>

a. If Message_Type is equal to 1, then Field_A correspond to Switch_ID and Field_B correspond to Switch_State (1 for on, 0 for off).
b. If Message_Type is equal to 2, then Field_A correspond to Sensor_ID and Field_B correspond to Sensor_Reading (between 0 and 255).
c. If message type is 10, then the message corresponding to Field_A is the number of switches and Field_B holds the number of sensors of a switchboard corresponding to switch_board_id.

5.2 Steps when a switch on switch board is turned on/off

1. Whenever a user opens the website, a persistent web socket is created between the web server and the web browser.
2. Whenever a user turns on/off a switch on the switchboard (which is connected to the Arduino), a message is passed from Arduino to Raspberry Pi server via XBee as shown in 5.1 a above.
3. The message is then analyzed by the Web Server which then updates the corresponding switch state in the MySQL database. The web server broadcasts the message to all the websockets which results in updation of the switch state on all open web pages.

5.3 Steps when a switch on website is turned on/off

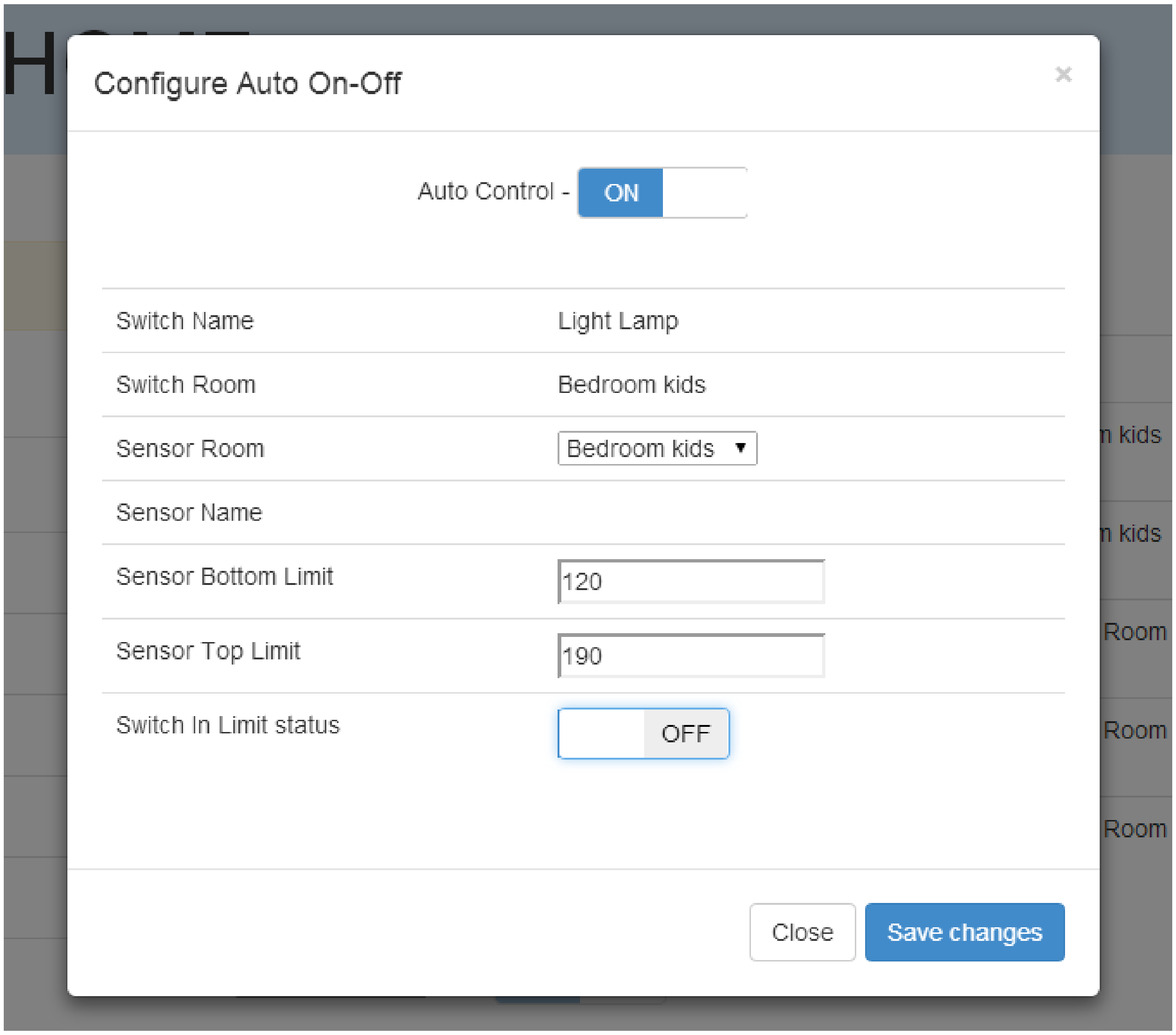
1. Whenever a user turns on/off a switch on the website, a message is passed from browser to web server through the web socket. The message contains the switch id, room id and the new switch state.
2. On receiving the message, server updates the switch info in the switches table in the database.
3. Server then broadcasts a command to the Arduino boards through XBee similar to the command as shown in **5.1 a**.
4. The Arduino board with the same <switch_board_id> as in the command will turn on/off the required switch according to the command.
5. Server also writes the message to all the sockets so as to synchronize the switch state in all browsers.

5.4 How are sensor values updated

1. Arduino sends messages of type **5.1 (b)** to Web server at regular intervals (currently set to 2 seconds) through XBee. These messages contains sensor readings.
2. The message is then analyzed by the Web Server which then updates the corresponding sensor value in the "sensors" table. The server also checks the switches which are auto configured (Refer to section 5.5) to some sensor and takes appropriate action.
3. Server also broadcasts the sensor values to all the sockets so as to synchronize the sensor values in all browsers.

5.5 Automatic on/off switch via Sensors

With each switch, we have added an option to auto control it. Switch can be bind to a sensor for auto on/off based on sensor value. On clicking the "Configure Auto" button in Figure 3, a modal opens up as shown in Figure 4. It asks for the sensor which we want to bind with the switch. It also asks for the range in which we need to turn on/off the switch and the default behavior in the range. On clicking "Save Changes" button, this information is sent to server which updates it into table "sensor_switch_maps" in MySQL database.


Configure Auto On-Off

Auto Control - ON

Switch Name	Light Lamp
Switch Room	Bedroom kids
Sensor Room	Bedroom kids ▾
Sensor Name	
Sensor Bottom Limit	120
Sensor Top Limit	190
Switch In Limit status	OFF

Figure 4 : Auto Configuring Switches

5.6 Discovery Protocol:

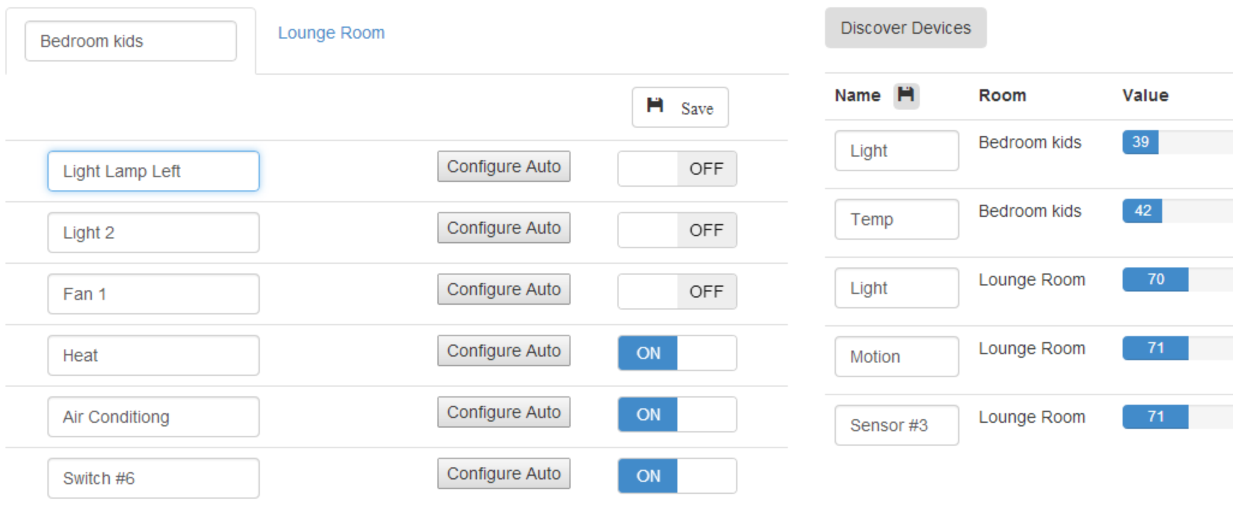
1. Whenever the user clicks the button "Discover Devices" the socket connection sends out a message to the server from the web page. This leads to the server broadcasting a message to arduinos via XBee of type 5.1-c (with id 10).
2. When the arduino receives this message it replies back with messag of type 5.1-c with the fields filled in with the number of switches and sensors.
3. On receiving the message sent out by Arduinos the server updates the respective mysql tables with distinct and default names given to each sensor and switch.
4. This should be ideally run after each installation of a new board, switch or sensor.

5.7 Assigning Names to Switches, Sensors and Rooms:

After the discovery protocol has been run, the user can assign names to each of the switches and rooms discovered. This is shown in Figure 5. The steps are as follows-

1. On Clicking the edit button provided at the top of the rooms tab all the fields become writable.
2. The user can then go on to edit these names according to his/her convenience. Once edited these can be saved at the click of a button, which initiates communication to the web server (RaspberryPi) over NodeJs sockets.
3. Finally these names are updated in the MySQL database on the server.

Similar steps follow in the renaming of sensors, which can be initiated by clicking the edit button on the top of the sensors table.



Light Lamp Left	Configure Auto	OFF
Light 2	Configure Auto	OFF
Fan 1	Configure Auto	OFF
Heat	Configure Auto	ON
Air Conditiong	Configure Auto	ON
Switch #6	Configure Auto	ON

Name	Room	Value
Light	Bedroom kids	39
Temp	Bedroom kids	42
Light	Lounge Room	70
Motion	Lounge Room	71
Sensor #3	Lounge Room	71

Figure 5 : Editing Room names, Switch Names and Sensor names

5.8 Test Results:

We tested the system by opening multiple web pages and attaching two arduino switch board. We also attached a light sensor to check the use of sensors to switch values on or off. All these tests yielded a maximum delay of 2 seconds between actions on the board to the web page and vice versa. This can be clearly seen in the demonstration video.

6. Discussion of System

a) What all components of your project worked as per plan?

All except one components worked as per plan. We had resolved to implement Relay driving circuits in the SRS document but ended up not doing it both due to time constraints and implementation difficulty.

b) What functionalities were added other than those discussed in SRS?

1. **Sensor Readings:** We felt the need to add sensors to the system. So, we added sensors to switch boards, which allow us to see real time sensor values on the website. Sensors can be of different types like light sensor, temperature sensor, etc.
2. **Automatic Switch On/Off via sensor readings:** To make the system smarter, we added the functionality to automatic switch on/off the switches based on sensor readings. Through this, a user can bind a switch with a particular sensor and specify the sensor range in which he/she

wants to turn on/off the switch. So, when the specified sensor reading is within the specified range, the system will automatically turn on/off (as specified in the settings) the switch.

c) Changes made in plan from SRS:

Initially, We decided to use Ajax and PHP for making client-server application. We implemented it but it seemed very slow and we needed a faster system for real time communication. Then we decided to go for NodeJS. NodeJS allows real time client-server communication by making persistent web sockets. We also couldn't implement the Relay Driving circuits.

7. Future Work

We have mainly two components: web server hosted on the Raspberry Pi and Switch Board attached with the Arduino. Both of these components are re-usable. If one can understand the working of this system correctly, then either of these components can be combined with similar other components. For example, the same web server hosted on the same Raspberry pi connected to the same XBee module can be used with another switch board designed by someone else just by following the same communication protocol.

Possible extensions of the project:

1. **Scheduling :** A scheduling algorithm can be implemented on the server to schedule events like turning on/off switches according to the time, turning on the light when you return from home is one example for this use case.
2. **Addition of a video/audio feed:** We could provide a modal with the video feed of the home or greenhouse to monitor the location for safety and/or maintenance purposes. we could also similarly add audio feed if required.
3. **Use of Mux/Demux:** As arduino has limited number of pins, multiplexer and demultiplexer can be used to increase the number of switches attached to arduino.

7.1 Bugs

A part of future work could be resolving the bugs that exist in the system. The only bug in the present system is that the page needs to be refreshed before the sensor name is reflected on all other web pages where the socket connection exists, this is not an issue in the case of rooms and switches or switch-sensor maps.

8. Conclusions

We believe that the project has achieved ease of switching electrical connections for the user, in a cost effective and efficient way. The control of the switches using the sensors ensures the smartness of the system as a whole. The real time data delivery using NodeJs was a challenging affair to implement resulting in a real time responsive system. The circuitry used was designed and implemented with the goal that a good demonstration of the system could be given, but a need for using relay driving circuits. The communication between XBees by defining our own commands and protocols was a challenging task. All these can be put to use in future work.

The present system can be used to switch electrical fixtures either manually or with sensor constraints. The discovery protocol has been implemented to provide the user the ease of attaching new sensors and switches without changing Arduino code. The use of the edit button to change names has been implemented to ensure that no code changes are needed on the Raspberry Pi server. The circuitry described above can be used to make relevant connections in a real system being deployed.

Overall this project has been a great learning experience for all the team members. We have learnt a lot regarding how an Embedded System is different from any other software developement project. We have also learnt different technologies in the process. In addition to this we have also learnt the stages of prototyping a product in an entrepreneurial environment and gotten a peek into how projects are dealt with when there is an end user involved.

9. References

- NodeJS: http://nodejs.org/
- Jade Template Library: http://jade-lang.com/
- Bootstrap CSS Framework: http://getbootstrap.com/
- Ajax Javascript Library: https://api.jquery.com/jQuery.ajax/
- Arduino Programming Language: http://arduino.cc/en/Reference/HomePage#.UxB-r89YKgg